



<b>Experiment No.4</b>
<b>Implementation of Queue menu driven program using arrays</b>
<b>Name: Siddhi Gaikwad</b>
<b>Roll No: 10</b>
<b>Date of Performance:</b>
<b>Date of Submission:</b>
<b>Marks:</b>
<b>Sign:</b>

#### **Experiment No. 4: Simple Queue Operations**

**Aim: To implement a Linear Queue using arrays.**

**Objective:**

- 1 Understand the Queue data structure and its basic operations.
2. Understand the method of defining Queue ADT and its basic operations.
3. Learn how to create objects from an ADT and member functions are invoked.

**Theory:**

A queue is an ordered collection where items are removed from the front and inserted at the rear, following the First-In-First-Out (FIFO) order. The fundamental operations for a queue are "Enqueue," which adds an item to the rear, and "Dequeue," which removes an item from the front.



**(b) A computer queue**



## **Vidyavardhini's College of Engineering and Technology**

### **Department of Artificial Intelligence & Data Science**

So, the output of evaluating the postfix expression "AB+C-" is "A + B - C."

This input would be accepted by a program designed to evaluate postfix expressions, and the output would be "A + B - C."



```
/tmp/Uux1dFb17F.o
ASSUMPTION: There are only four operators(*, /, +, -) in an expression
and operand is single digit only.

Enter postfix expression,
press right parenthesis ')' for end expression : (1
+56/-46+)stack under flow
Result of expression evaluation : 10
```

### **Conclusion:**

Elaborate the evaluation of the following postfix expression in your program.

AB+C-

Will this input be accepted by your program. If so, what is the output?

To evaluate the postfix expression "AB+C-," follow these steps:

1. - When you encounter an operand (A), push it onto the stack.
- 2.- When you encounter another operand (B), push it onto the stack.
3. When you encounter an operator (+), pop the top two operands from the stack (B and A in this case), perform the addition ( $A + B$ ), and push the result back onto the stack.
4. Continue scanning the expression.
5. When you encounter the next operator (-), pop the top two operands from the stack (result of  $A + B$  and C in this case), perform the subtraction ( $A + B - C$ ), and push the final result back onto the stack.
6. After processing the entire expression, the final result is the only item left in the stack, which is " $A + B - C$ ."



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
case '+': val = B + A;

break;

case '-': val = B - A;

break;

    } push(val); }

    } printf(" \n Result of expression evaluation : %d \n", pop());

} int main() {

    int i;

    char postfix[POSTFIXSIZE];

    printf("ASSUMPTION: There are only four operators(*, /, +, -) in an expression and operand
    is single digit only.\n");

    printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

    for (i = 0; i <= POSTFIXSIZE - 1; i++) {

        scanf("%c", &postfix[i]); if (postfix[i] == ')')

        {

            break;

        }

    } EvalPostfix(postfix);

    return 0;

}
```

**Output:**



## **Vidyavardhini's College of Engineering and Technology**

### **Department of Artificial Intelligence & Data Science**

Typically, a one-dimensional array is used to implement a queue, and two integer values, FRONT and REAR, track the front and rear positions in the array. When an element is removed from the queue, FRONT is incremented by one, and when an element is added to the queue, REAR is increased by one. This ensures that items are processed in the order they were added, maintaining the FIFO principle.

#### **Algorithm:**

ENQUEUE(item)

1. If (queue is full)

    Print "overflow"

2. if (First node insertion)

    Front++

3. rear++

Queue[rear]=value

DEQUEUE()

1. If (queue is empty)

    Print "underflow"

2. if(front=rear)

    Front=-1 and rear=-1

3. t = queue[front]

4. front++

5. Return t

ISEMPTY()

1. If(front = -1)then

    return 1

2. return 0



**Vidyavardhini's College of Engineering and Technology**  
**Department of Artificial Intelligence & Data Science**

ISFULL()

1. If(rear = max)then

return 1

2. return 0

**Code:**

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
#define MAX 10
```

```
// Changing this value will change length of array
```

```
int queue[MAX];
```

```
int front = -1, rear = -1;
```

```
void insert(void);
```

```
int delete_element(void);
```

```
int peek(void);
```

```
void display(void);
```

```
int main()
```

```
{ int option, val;
```

```
do {
```

```
printf("\n\n ***** MAIN MENU *****");
```

```
printf("\n 1. Insert an element");
```

```
printf("\n 2. Delete an element");
```

```
printf("\n 3. Peek");
```

```
printf("\n 4. Display the queue");
```

```
printf("\n 5. EXIT");
```



**Vidyavardhini's College of Engineering and Technology**  
**Department of Artificial Intelligence & Data Science**

```
printf("\n Enter your option : ");  
  
scanf("%d", &option);  
  
switch(option)  
{  
    case 1:  
        insert();  
        break;  
    case 2:  
        val = delete_element();  
        if (val != -1)  
            printf("\n The number deleted is : %d", val);  
        break;  
    case 3:  
        val = peek(); if (val != -1)  
            printf("\n The first value in queue is : %d", val);  
        break;  
    case 4:  
        display();  
        break;  
}  
}  
  
while(option != 5);  
  
getch();  
  
return 0;  
}  
  
void insert() {
```



**Vidyavardhini's College of Engineering and Technology**  
**Department of Artificial Intelligence & Data Science**

```
int num;

printf("\n Enter the number to be inserted in the queue : ");

scanf("%d", &num);

if(rear == MAX-1)

printf("\n OVERFLOW");

else if(front == -1 && rear == -1)

front = rear = 0;

else

rear++;

queue[rear] = num;

}

int delete_element() {

int val;

if(front == -1 || front>rear) {

printf("\n UNDERFLOW");

return -1;

} else {

val = queue[front];

front++;

if(front > rear)

front = rear = -1;

return val;

}

}

int peek() {

if(front== -1 || front>rear) {
```





## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
printf("\n QUEUE IS EMPTY");  
  
return -1;  
  
}  
  
else {  
  
return queue[front]; }  
  
}  
  
void display() {  
  
int i;  
  
printf("\n");  
  
if(front == -1 || front > rear)  
  
printf("\n QUEUE IS EMPTY");  
  
else {  
  
for(i = front;i <= rear;i++)  
  
printf("\t %d", queue[i]);  
  
}  
  
}
```

#### Output:

```
***** MAIN MENU *****  
1. Insert an element  
2. Delete an element  
3. Peek  
4. Display the queue  
5. EXIT  
Enter your option : 1  
Enter the number to be inserted in the queue : 50
```



# **Vidyavardhini's College of Engineering and Technology**

## **Department of Artificial Intelligence & Data Science**

### **Conclusion:**

What is the structure of queue ADT?

The Queue Abstract Data Type (ADT) is a linear data structure that follows the First-In-First-Out (FIFO) principle, meaning that the first element added to the queue is the first one to be removed. The basic operations and characteristics of a queue ADT include:

1. Enqueue: Adds an element to the back (rear) of the queue.
2. Dequeue: Removes and returns the element from the front (head) of the queue.
3. Peek (or Front): Allows you to view the element at the front of the queue without removing it.
4. IsEmpty: Checks if the queue is empty.
5. Size: Returns the number of elements currently in the queue.

A simple real-life analogy for a queue is a line of people waiting for a service, where the person who arrives first is the first to be served.

List various applications of queues?

Here are various applications of queues:

1. Print Queue: Managing print jobs in order.
2. Task Scheduling: Scheduling tasks in operating systems.
3. Breadth-First Search: Traversing graphs level by level.
4. Call Center Systems: Handling customer service calls.
5. Buffer Management: Storing data in a temporary buffer.



## **Vidyavardhini's College of Engineering and Technology**

### **Department of Artificial Intelligence & Data Science**

6. Request Handling: Managing requests in web servers.
7. Bounded Buffer: Handling data between producers and consumers.
8. Job Scheduling: Scheduling tasks in computing clusters.
9. Simulation Systems: Modeling real-world scenarios.
10. Task Queues: Managing background tasks in applications.

Where is queue used in a computer system processing?

Queues are used in a computer system for tasks like:

1. Task scheduling in operating systems.
2. Managing I/O operations.
3. Handling print jobs.
4. Efficient interrupt handling.
5. Buffer management.
6. Job queues in computing clusters.
7. Synchronization.
8. Message queues in distributed systems.
9. Request handling in web servers.
10. Background task management in applications.

They ensure orderly and efficient processing of tasks, data, and requests.